|  |
| --- |
| gbXML Implementation Guidelines |
| For Geometry Definitions Required for Validation |
| Chien Harriman |



Contents

[Introduction 2](#_Toc347215697)

[General Overview - How Geometry is Described in gbXML 2](#_Toc347215698)

[1. ShellGeometry Node Basics 2](#_Toc347215699)

[2. SurfaceBoundary Node Basics 3](#_Toc347215700)

[3. Surface Node Basics 3](#_Toc347215701)

[Examples of a Simple Surface Defined by Shell Geometry and Space Boundary 3](#_Toc347215702)

[Example of a Simple Surface Defined by the Surface Element 10](#_Toc347215703)

[A High-level overview of the Surface element’s Insertion Point Convention 15](#_Toc347215704)

[Anatomy of a Surface Element 11](#_Toc347215705)

[Definitions of Important Surface Element Attributes and Descendants 12](#_Toc347215706)

[Further Insertion Point Definitions 15](#_Toc347215707)

[Interior Walls 15](#_Toc347215708)

[InteriorFloors(Ceilings) 16](#_Toc347215709)

[Roofs 17](#_Toc347215710)

[Example 1 – A Simple Cube Completely Defined in gbXML 18](#_Toc347215711)

[Roof is Constructed with all insulation thickness above grade 18](#_Toc347215712)

[Special issues re: ShellGeometry and SpaceBoundary Elements 23](#_Toc347215713)

[Special issues re: Surface Elements 27](#_Toc347215714)

**gbXML Implementation Guide**

# Introduction

gbXML is a data transfer format optimized to contain information relevant to a building and all of its parts that impact its energy usage. The structure and format of the language is governed by industry stakeholders who oversee updates and use cases that test the strength of gbXML and its intended purpose.

This agreement is intended for software engineers, engineers, architects, and other interested parties wish to understand methods for defining geometric objects in gbXML. All of those components that make up the elements of a room or space… covering the description of surface locations and thicknesses of *orthogonal* volumes.

This agreement covers the cases where the BIM authoring tool exports *either* surfaces with thickness *or* surfaces that have no thickness (aka – egg shell geometry), and what is to be expected in the gbXML file in each case. We make this distinction becauseBIM authoring tools differ in their description of geometry on this basis. It is expected that elements with no thickness will be described differently than those elements that do have a thickness, and it is important to cover both cases.

Interior and exterior surfaces must be defined by some form of geometric description referenced to a global reference frame. In this implementation guide, it is implied that the coordinates of the points that describe these surfaces are defined relative to this global reference frame. When encoded to gbXML, these coordinates must be properly translated into the proper format with the appropriate meta data as per the latest gbXML XSD.[[1]](#footnote-1) In other words, the goal of this agreement is not only to ensure that the BIM authoring tool is capable of merely translating the coordinates from its internal global reference frame to a gbXML file. This document clarifies how the surfaces of a building shall be properly formatted given these coordinates, so that surfaces are properly defined and consistently well-formed by the authoring tools when translated into gbXML.

# General Overview - How Geometry is Described in gbXML

gbXML files use three different parent XML nodes (which also have children associated with them). You may review the hierarchy of these nodes in the latest gbXML XSD. The word node and element are used interchangeably throughout this document.

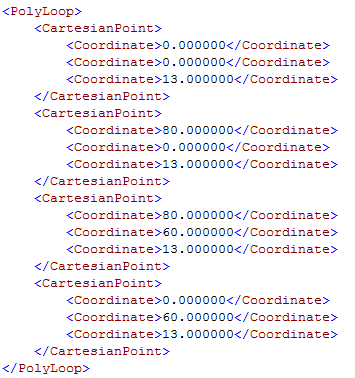
1. ShellGeometry Node
   1. defines *the inner surface* of a Space (sometimes referred to as the wetted or interior surface)
   2. is a direct descendant of a Space node, so its relationship to the space is implied by its parent-child relationship inherent in the XML schema
2. SpaceBoundary Node
   1. defines the *centerline* of a surface bounding Space
   2. is a direct descendant of a Space node, so its relationship to the space is implied by its parent-child relationship inherent in the XML schema
3. Surface Node
   1. defines the surface of a Space, with geometry characteristics similar to the SpaceBoundary
   2. defines surfaces that are not contained within spaces, such as Shading Devices
   3. defines openings in Surfaces (windows, skylights, doors, etc.), which is unique to these types of nodes
   4. is not a direct descendant of a Space node, therefore its relationship to the space must be alternatively defined within its attributes or children. Use of AdjacentSpaceId is the means by which surfaces are bound to spaces.

Most engineering tools that consume gbXML do not consume all three different geometry descriptions. In general, it has been found without exception that tools will use a combination of ShellGeometry and SpaceBoundary elements or will use the Surface elements only to gather the geometric information necessary to convert the gbXML coordinate data into a geometrically accurate representation of the building.

Software tools that use both ShellGeometry and SpaceBoundary are interested in the information contained therein, since the combination of this information allow a more accurate representation of the surface, including an accurate representation of the the surface thickness. This may be more ideal for tools that need this thickness for accuracy…daylight of CFD simulations. Many tools prefer the Surface element representation; it is simpler and is suitable for most energy and load simulation programs.

# Examples Showing the Requirements for a Simple Surface Defined by Shell Geometry and Space Boundary Elements

ShellGeometry and SpaceBoundary elements shall be defined purely with geometric coordinates defined in a Cartesian reference frame. The reference frame assumes that the +Y direction is true north, the +X direction is East, and the +Z direction is towards the sky. The coordinates are contained within a <PolyLoop></PolyLoop> node, which in turn contains a nested node that contains the X,Y,Z coordinates of each point that defines the surface.



The orientation of the walls is not explicitly declared in any child node or attribute of a ShellGeometry element nor any SpaceBoundary element. Instead the orientation of the surface (which side is pointing outwards and inwards, which is crucial for energy analysis. is implied by the normal vector that is formed by right hand rule that can be calculated from creating vectors sequentially with the coordinates. Thus the order of the Cartesian coordinates in a PolyLoop are incredibly important. There are no requirements as to which coordinate of a surface must come first in this sequence of coordinates, though some tools that consume gbXML do declare which coordinate should be first in the sequence. In short, it should be remembered that both the values of the Cartesian coordinates, *and the order* in which they are written, define a properly defined ShellGeometry and SpaceBoundary element.

The algorithm to determine orientation of a surface described in gbXML uses a right hand rule calculation methodology. The algorithm takes the coordinates that it finds sequentially, forming vectors with these coordinates, and then taking the cross product of these resultant vectors. Here is a simple example explaining how the order of the coordinates is important, given this simple explanation of the algorithm for the right hand rule:

Figure : Right hand rule example diagram showing sequence of coordinates and the resulting normal vector.

In order for the right hand rule calculation to result in a vector pointing in the direction shown (for simplicity, we can declare the orientation east), then the coordinates must be sequenced in the manner identified by the arrows linking the sides of the shape. It does not matter which coordinate is specified first in the sequence, just so long as the sequence follows the same order or rotation. This may run counter to certain energy simulation programs that may be required to specify the location of the first coordinate in a list of coordinates.

To provide more specific examples, below is an excerpt from a valid gbXML file showing the ShellGeometry and SpaceBoundary geometric definitions for a simple rectangular wall. The actual value of the coordinates is not important, but to give a sense of the structure and the rules for valid gbXML.

The key information in a ShellGeometry node and its descendants is all contained in the PolyLoop and the CartesianPoint and Coordinate nodes. The id attribute must be unique, and the units attribute helps to identify the units of measure. For a complete description of all attributes for the ShellGeometry element and its descendants, please refer to the latest version of the gbXML XSD.[[2]](#footnote-2)

## High level rules for ShellGeometry Elements and its Descendants:

* each PolyLoop defines a surface of a Space
* a ShellGeometry element may consist of an unlimited number of PolyLoops
* each PolyLoop node may have an unlimited number of CartesianPoint nodes as children
* the PolyLoop set of coordinates may start with any Coordinate in the set that defines the plane
* the Coordinate elements must form a planar, non self-intersecting polygon when the points are read in order from first to last
* the CartesianPoint elements must be in proper order for right hand rule verification
* each CartesianPoint will have three Coordinate elements (order is important)
  + 1st coordinate is the *implied* X coordinate in the BIM authoring tool global ref
  + 2nd coordinate is the *implied* Y coordinate in the BIM authoring tool global ref
  + 3rd coordinate is the *implied* Z coordinate in the BIM authoring tool global ref

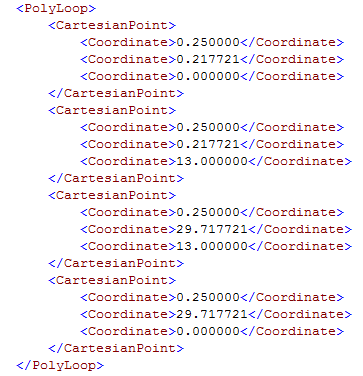


<ClosedShell>

<PolyLoop>

….some Cartesian points, e.g.

</PolyLoop>



</ClosedShell>

</ShellGeometry>

Code Snippet : Anatomy of a ShellGeometry Element.

In the example above, we have shown a ShellGeometry element with only two PolyLoop elements, which in practice is not feasible because this would not be enough surfaces to describe any volume in Cartesian space. However, this is just an illustration. Note that the ShellGeometry element can contain more than one (and in fact an unlimited quantity of) PolyLoops. We have expanded the description of one of the PolyLoop elements here to clarify how CartesianPoint elements and Coordinate elements are defined.

Though not a requirement of many engineering tools that consume gbXML, ShellGeometry PolyLoop coordinates should, when analyzed by a computer algorithm, form a watertight volume. In other words, each coordinate should be replicated at least once in another surface, to form a perfectly enclosed volume. An algorithm should be able to determine whether the surfaces do indeed create an enclosed volume, or not.

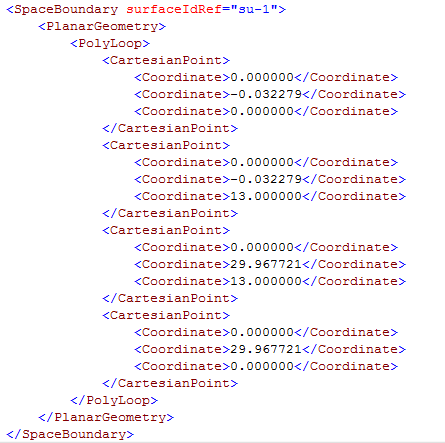
In each Space element, after the ShellGeometry closing tag, the SpaceBoundary is next defined. The key information in a SpaceBoundary node and its descendants is also contained in the PolyLoop and the CartesianPoint and Coordinate nodes, like its counterpart the ShellGeometry. The surfaceIdRef attribute is also important, and though it is not required that the value of this attribute be unique, there are rules that govern how it is used. For a complete description of all attributes for the SpaceBoundary element and its descendants, please refer to the latest version of the gbXML XSD.[[3]](#footnote-3)

## High Level Rules for SpaceBoundary Elements and its Descendants.

* There shall be the same number of SpaceBoundary elements as ShellGeometry PolyLoops for each Space element.
* A human should be capable of matching a given SpaceBoundary PolyLoop to its ShellGeometry PolyLoop counterpart

The SpaceBoundary also has descendant nodes called PolyLoop. This PolyLoop and its descendants follow rules similar to the PolyLoop rules defined for the ShellGeometry PolyLoop shown in the code excerpt above.

* there is only one PolyLoop per SurfaceBoundary (different than ShellGeometry PolyLoops, which may be unlimited in number)
* each PolyLoop may have an unlimited number of CartesianPoint elements (same as ShellGeometry PolyLoops)
* the PolyLoop set of coordinates may start with any Coordinate in the set that defines the plane
* each CartesianPoint will have three Coordinate nodes (same as ShellGeometry PolyLoops)
  + 1st coordinate is the *implied* X coordinate in the BIM authoring tool global ref
  + 2nd coordinate is the *implied* Y coordinate in the BIM authoring tool global ref
  + 3rd coordinate is the *implied* Z coordinate in the BIM authoring tool global ref
* the CartesianPoint elements must be in the proper order for right hand rule verification (same as ShellGeometry PolyLoops)



Code Snippet : Anatomy of a Space Boundary Element.

It was previously mentioned that it should be possible to logically match a SpaceBoundary PolyLoop to a ShellGeometry PolyLoop. The two do bear a relationship to one another. Whereas the ShellGeometry element defines the inner surface of a wall or floor or ceiling, the SpaceBoundary defines the location of the centerline of that surface. It is in this way that tools that require knowledge of the thickness of the surface can deduce it by comparing the coordinate geometry of a ShellGeometry element and its complimentary SpaceBoundary element.

## High Level Rules for Relationships between ShellGeometry Elements, SpaceBoundary Elements, and Their Descendants

* Regardless of whether a surface is an interior or exterior surface, the ShellGeometry PolyLoop elements will always describe each surface in a Space.
* SpaceBoundary elements for surfaces that are exterior in nature (one side is adjacent to a space, but the other is adjacent to the Outdoors/Grade/Below Grade), are each individually defined and have a unique spaceIdRef.
* Interior surfaces (those that share a boundary on each side by a Space element), are not all individually defined by SpaceBoundary elements. These interior surfaces share the same spaceIdRef. To put this another way:
  + If two space boundaries share the same spaceIdRef, this means that this SpaceBoundary element defines an interior surface (one that separates one space from another).

# 

# Example of a Simple Surface Defined by the Surface Element

Surface elements, as opposed to ShellGeometry and SpaceBoundary elements, are more compact descriptions of surfaces. Whereas the ShellGeometry and SpaceBoundary coordinate geometry must be analyzed to determine the relative position of a given surface described by those elements, i.e. – the coordinates when analyzed reveal which surfaces are exterior walls, interior walls, slab on grade floors, roofs, and other surfaces, etc… Surface elements are defined by its surfaceType attribute, an insertion point, reference to an adjacency relationship, Tilt, Azimuth, Width, Height, and an optional PolyLoop that defines the coordinates for that surface. In this sense, the Surface element is a ‘pre-baked’ description of a surface that greatly simplifies the work of determining a surface orientation and adjacency relationships, without requiring a complex analysis of PolyLoop coordinates to determine location (as would be needed if ShellGeometry and SpaceBoundary elements only were provided). This makes Surface elements the primary choice for most energy analysis tools to convert gbXML data into energy simulation or loads calculation data.

Surface elements are defined by a surfaceType attribute, that declares the Surface to be one of 12 (twelve) different building surface categories. These categories are enumerations held by the surfaceTypeEnum declaration in the gbXML XSD.

* surfaceType = “ExteriorWall”
* surfaceType= “UndergroundWall”
* surfaceType = “InteriorWall”
* surfaceType = “Air”
* surfaceType = “InteriorFloor”
* surfaceType = “UndergroundSlab”
* surfaceType = “RaisedSlab”
* surfaceType = “Ceiling”
* surfaceType = “UndergroundCeiling”
* surfaceType = “Roof”
* surfaceType = “Shade”
* surfaceType = “FreestandingColumn”
* surfaceType = “EmbeddedColumn”

Once this type is declared, the Surface simply needs some simple geometric information and relationship definitions to be defined. See the example below for the Surface element definition of the surface described by the ShellGeometry and SurfaceBoundary elements in the example above.

## Anatomy of a Surface Element

<Opening>

…optional openings defined here

</Opening>

….may contain multiple openings whose total area does not exceed the area of the surface

Code Snippet : Surface Element Description.

</Surface>

Let us now explain the various components of a Surface element. The optional Opening child element will be described in more detail below.

## Definitions of Important Surface Element Attributes and Descendants

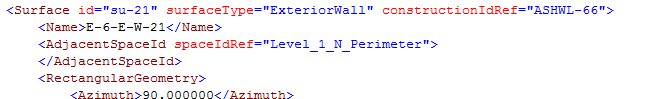
id: contains a string that is identical to the value of the SurfaceBoundary element attribute surfaceIdRef

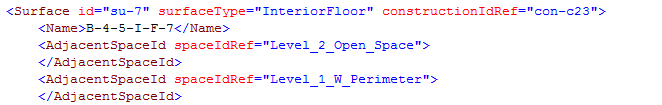
surfaceType: value is one of surfaceTypeEnum values declared in the latest XSD (Refer for complete list)

constructionIdRef: reference to a construction that will be exported and associated with the surface. The details of this construction are found in the \_\_ element in the gbXML file. More information can be found in the latest XSD file.

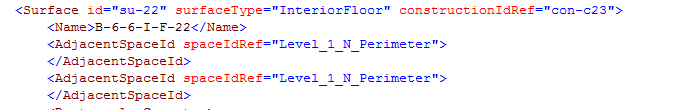
Name: a unique string that identifies the element

AdjacentSpaceId: an element that may be repeated up to twice for each Surface. If repeated only once, the gbXML convention is that this Surface is an exterior surface. The value of the attribute spaceIdRef is identical to the value of the id attribute of the associated Space element.



If repeated twice for a given Space element, and the surfaceTypeEnum is “Interior \_\_\_”, then the values of the spaceIdRef attributes for each AdjacentSpaceId shall not be identical. The first unique instance of AdjacentSpaceId element holds a value in attribute spaceIdRef that is one of the two Spaces adjacent to this Surface. The second unique instance holds the value of the second Space that is adjacent to this Surface. Example:

If the AdjacentSpaceId element is repeated twice for a single Surface element, and both instances of the spaceIdRef attribute contains a value whose string is identical to the other, and the surfaceTypeEnum is “InteriorFloor”, then this implies the floor surface in question is slab on grade. Example:



Azimuth: the rotation of the surface from project true north. It is the direction of the outward normal as calculated by the right hand rule of the PolyLoop (described below). It is an angular measure in degrees clockwise from true north, where 0 degrees = north pointing. This angle only measures rotation in the X-Y plane.

If CADModelAzimuth is defined, then the azimuth is relative to this value rather than true north. In the figures below, both walls (outward normal defined by the arrow) would have an azimuth of 180. This is due to the fact that the CADModelAzimuth is set to 90o for the figure on the right, rotating the project north by 90o. Now the azimuth is no longer coinciding with true north, but project north.

Example Figure a: CADModel Azimuth = 0, Surface Azimuth = 180

True North

wall

Project North

Example Figure b: CADModelAzimuth = 90, Surface Azimuth = 180

True North

wall

Project North

Cartesian Point: Defines the insertion point for the surface. The CartesianPoint will have three Coordinate nodes

* + 1st coordinate is the implied X coordinate in the BIM authoring tool global ref
  + 2nd coordinate is the implied Y coordinate in the BIM authoring tool global ref
  + 3rd coordinate is the implied Z coordinate in the BIM authoring tool global ref

This document provides several examples of the conventions that govern the insertion point location below.

Tilt: the second angular measure that also is in reference to the direction of the outward normal. The angular measure is the angle between the outward normal and the z-axis in the BIM authoring tool’s global reference frame. It is always assumed that the Z-axis is pointing perpendicular to the ground plane, up towards the sky.

* Tilt = 0-45 is a flat roof
* Tilt = 45-135 is a wall
* Tilt = 135-180 is a floor

Height: the height of a surface. This value, along with Width, is the easiest way to calculate the area of the surface. For surfaces that are orthogonal quadrilaterals, the height can be used along with the insertion point CartesianPoint and Width to reconstruct the optional PolyLoop Coordinate elements. If the surface is not an orthogonal quadrilateral, the height can still be entered as a quick means to calculate area, though in reality it does not represent the true shape or actual height of the object, it is a simple transformation from the real shape and height to the height of an orthogonal quadrilateral that is thermodynamically equivalent. See the discussion on thermodynamic equivalence for more explanation.

Width: the width of a surface. This value, along with Height, is the easiest way to calculate the area of the surface. For surfaces that are orthogonal quadrilaterals, the width can be used along with the insertion point CartesianPoint and Height to reconstruct the optional PolyLoop Coordinate elements. If the surface is not an orthogonal quadrilateral, the width can still be entered as a quick means to calculate area, though in reality it does not represent the true shape or actual width of the object, it is a simple transformation from the real shape and width to the width of an orthogonal quadrilateral that is thermodynamically equivalent. See the discussion on thermodynamic equivalence for more explanation.

PolyLoop: each CartesianPoint will have three Coordinate nodes

* + 1st coordinate is the implied X coordinate in the BIM authoring tool global ref
  + 2nd coordinate is the implied Y coordinate in the BIM authoring tool global ref
  + 3rd coordinate is the implied Z coordinate in the BIM authoring tool global ref

the CartesianPoint elements must be in the proper order for right hand rule verification.

PolyLoop coordinates shall always be in the Surface definition if the Surface is being verified by the gbXML validator.

The PolyLoop is the only means to accurately describe the shape of a complex polygon that defines a planar surface.

## A High-level overview of the Surface Element’s Insertion Point Convention

The insertion point is a CarteisanPoint defined in the global reference frame. The insertion point is defined based upon an imagined observer’s point of view.

### surfaceType = ExteriorWall, UndergroundWall

For Exterior surfaces, when the surfaceType = “ExteriorWall” or “UndergroundWall”, the imagined observer is on the exterior side of the wall, looking in the anti-parallel direction to the wall’s outward facing normal, with the observer’s head pointing in the positive Z direction.

Figure : External Wall Insertion Point Definition

+z

Surface Normal

insertion point

Interior

Exterior

In this case, the insertion point is in the lower left hand corner identified by the large blue dot in the Figure above.

### surfaceType = InteriorWall. Air

For Interior surfaces, where the Surface element attribute surfaceType=”InteriorWall” or “Air”, there are two AdjacentSpaceId elements each with a unique value for its respected spaceIdRef attribute. *The second AdjacentSpaceId in the sequence will possess the value of the Space id attribute in which the observer is located*. The user then faces the outward normal looking in the anti-parallel direction to the normal vector of the wall and can observe the lower left corner that defines the origin of the insertion point. The observer’s head is pointed in the +z direction.

Adjacent

Space 2

Surface Normal

Adjacent

Space 1

Figure : Internal Wall

+z

origin

For interior surfaces, where the Surface element attribute surfaceType=”InteriorFloor”, there are

### surfaceType = InteriorFloor, UndergroundSlab, RaisedSlab

For interior surfaces where the Surface element attribute surfaceType=”InteriorFloor”. there are two AdjacentSpaceId elements each with the identical value for its respected spaceIdRef attribute. The insertion point for this type of surface is located in the lower left corner, assuming the observer is beneath the interior floor, lying on his back, with his head pointing towards true north. Ceilings may be defined in many cases by the BIM authoring tools with the Surface element attribute surfaceType=”InteriorFloor “.

Slab on grade is also defined in gbXML as an surfaceType=”InteriorFloor”. The Surface element’s constructionType attribute will provide the details of construction layers that translate the surface into a slab on grade, as opposed to a simple floor slab that separates floors in a multi-story building.

Figure : Interior Floor (including Slab on Grade Definitions)

origin

Surface

normal

+Z

N

### 

### surfaceType=Roof, Ceiling, UndergroundCeiling

For exterior surfaces, where the Surface element attribute surfaceType=”Roof”, there is one AdjacentSpaceId element each with a single value for its spaceIdRef attribute. The insertion point for this type of surface is located in the lower left corner, assuming the observer is above this name space, looking down on the roof, as if he were lying on his stomach, with his head pointing towards true south.

origin

+z

Surface

normal

S

# Example 1 – A Simple Cube Completely Defined in gbXML

## Roof is Constructed with all insulation thickness above wall height

To explain how gbXML is formatted to describe building surfaces, we shall take a simple example, a simple cube placed on grade. For BIM authoring tools that define a wall thickness (i.e. – not an egg shell model, the geometry may will be represented as a series of conflated cubes. We shall see how this example changes for the egg shell version shortly.

wall interior surface

wall centerline

wall exterior surface

+z

N

x

Figure : A simple axonometric view of a cube with wall thickness 'x', represented as Shell Geometry (thin blue solid line) Surface Boundary/Surface Element(thin blue dotted lines) and the exterior surface of the shape (thick red line).

roof cap thickness

(insulation above the

top of the walls)

slab on grade

thickness

z = 10

Figure : Plan View of same cube as Figure 5

z = 0

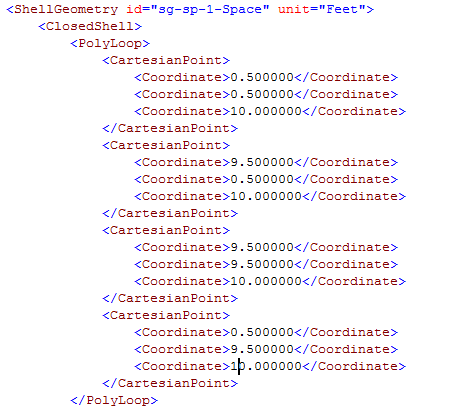
Figure : Section View of same cube as Figure 5

In the example above, imagine walls have been drawn in the BIM authoring tool with a thickness X, as shown in Figures 5-7. Each of the four walls is an exterior wall, with a roof capping the cube. The floor is slab on grade. The rules for defining these surface types area as follows

* For exterior walls
  + the ShellGeometry is defined at the inner surface of the wall thickness, where the PolyLoops that describe these elements will coincide with this inner wetted surface boundary. The SpaceBoundary and Surface Elements defined at the wall centerline, where the PolyLoops that describe these elements will coincide with the wall centerline. The exterior wall surface location is implied, since its coordinates can be deduced from the half distance that can be calculated by comparing the coordinate difference between the ShellGeometry coordinates and the SpaceBoundary elements.
  + The Surface element coordinates are defined at the centerline, just as the SpaceBoundary, though the thickness of the wall, whereas it was explicitly defined in the case of the SpaceBoundary, is instead only defined by the wall assigned to the Surface in the Surface element’s constructionIdRef attribute, independent of any relationship to the ShellGeometry PolyLoop coordinates
* For slab on grade
  + Note in Figure 7 that the thickness of slab on grade is not captured. The ShellGeometry, SpaceBoundary, and Surface elements and their descendants are all defined in the same plane.
* For Roofs
  + Note in Figure 7 that the thickness of a roof that is above the height of the exterior walls is not captured. The ShellGeometry, SurfaceBoundary, and Surface elements and their descendants are all defined in the same plane.
  + We will see an example of what happens when the inner surface of the roof is brought below the maximum height of a wall in a later example.

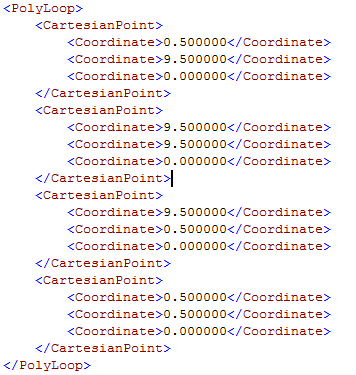
Note the special circumstances in Figure 7 where neither the slab-on-grade floor nor the Roof has any thickness captured. If the thickness of these objects were captured, the SpaceBoundary (blue dotted line) would extend to encompass this thickness, but it does not. Instead, it terminates at the same surface as the ShellGeometry. While in reality these two surfaces do have a thickness in reality and the BIM authoring tool, gbXML descriptions of these objects have historically been ignored. The thickness shall neither be represented by the ShellGeometry, SpaceBoundary, nor Surface Elements nor their descendants, respectively. The implications on the definitions of these three elements will be shown below in examples.

The gbXML representation for some of the representative surfaces for this cube is written below, assuming the cube is 10’ x 10’ x 10’ along the wall centerlines with 12” walls and a 12” roof whose insulation is completely above the height of the walls. Slab on grade is assumed to have the interior surface (ShellGeometry, SpaceBoundary, Surface elements) elevation of Z=0. Only one of the exterior walls has been defined for the sake of remaining brief in these descriptions.



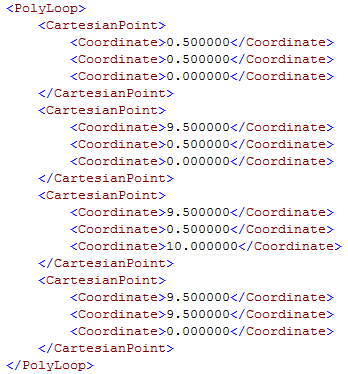
Roof

Code Snippet : Shell Geometry Polyloop for Roof in Example 1A.



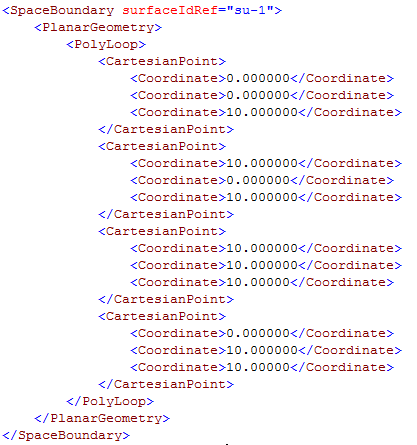
Code Snippet : Shell Geometry PolyLoop for Slab on Grade Interior Floor in Example 1A.

slab on grade



Code Snippet : Shell Geometry PolyLoop for the south facing Wall in Example 1A

South Wall



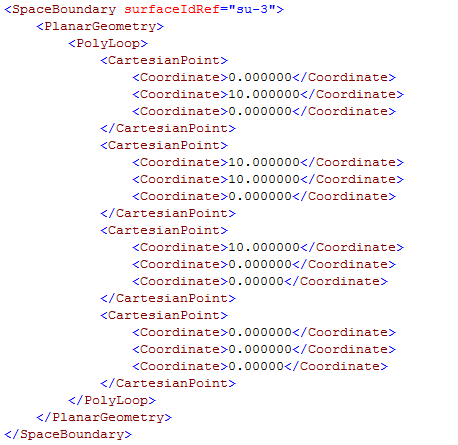
Roof

Code Snippet : Space Boundary definition for the Roof in Example 1A.



Code Snippet : SpaceBoundary definition for the South Wall in Example 1A.

South Wall



Slab on grade

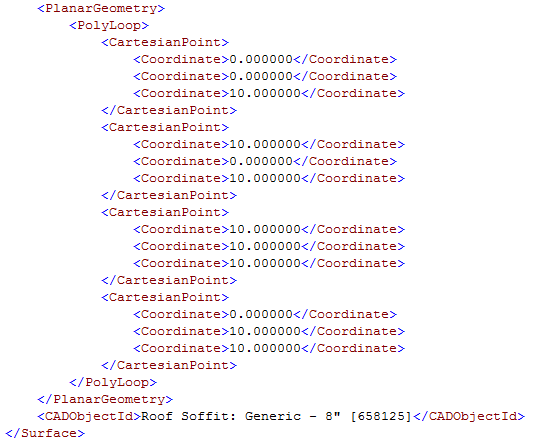
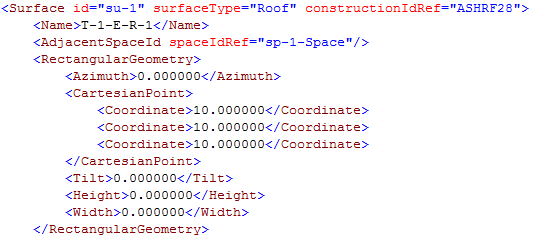
Code Snippet : SpaceBoundary definition for the Slab on Grade Floor in Example 1A.

### Special issues re: ShellGeometry and SpaceBoundary Elements

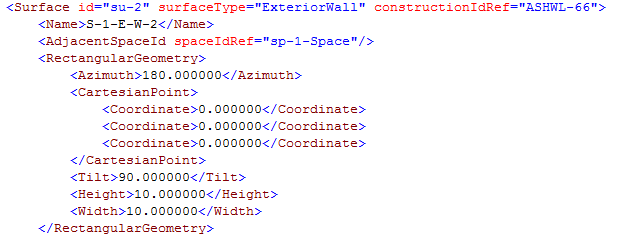
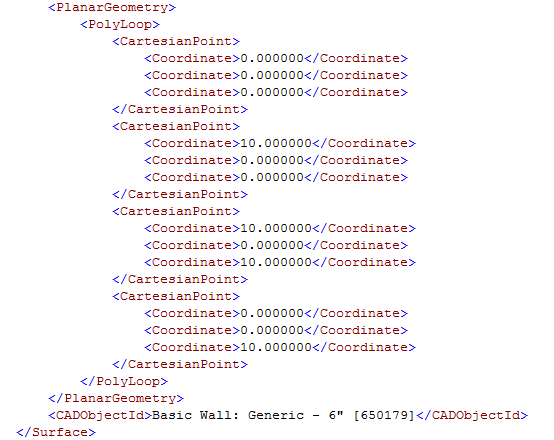
Of note from this example are:

1. The height of the wall is the same in both the ShellGeometry definition and the SurfaceBoundary definition.
2. The differences between the coordinate values are due to the thickness of the wall (12” total thickness, 6” half thickness). The origin of the project is at the centerline of the South and West Wall intersection, hence the shift by 0.5’ (6”).
3. The CartesianPoint elements are all sequenced to allow the proper calculation of the right hand rule.
4. The surfaceIdRef values each contain a number, which is the order in which the SpaceBoundary elements are created. The order does not need to match the order in which the ShellGeometry PolyLoop elements are defined.

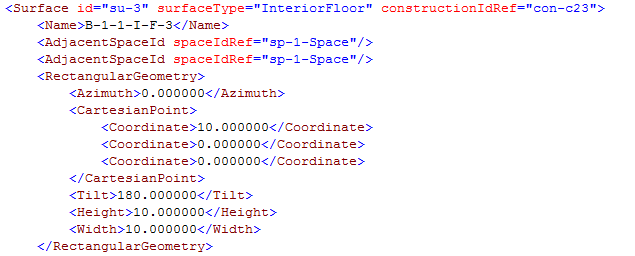
Surface definition examples below:



Code Snippet : Surface element definition for the Roof in Example 1A.

Code Snippet : Surface element definition for the South Wall in Example 1A.

Code Snippet : Surface element definition for the Slab on Grade Floor in Example 1A.

### 

### Special issues re: Surface Elements

1. The Surface element attribute id has a value that is identical to the SpaceBoundary surfaceIdRef. This is intentional.
2. A quick check would also reveal that the PolyLoop CartesianPoint elements are the same as the Polyloop CartesianPoint elements in the SpaceBoundary element, and sequenced in the same fashion. The order does not need to match exactly, but the sequence must be in the proper order for the right hand rule to be properly performed.

### Resulting PolyLoops Vectors

Figure : All resulting normal vectors for exterior surfaces or slab on grade, when calculated from the PolyLoop shall point outward.

For all PolyLoop elements in the ShellGeometry, SpaceBoundary, and Surface elements, the resulting normal vector calculated by processing the sequence of coordinates, should point outward, as shown in the figure above.

### How would an egg shell model be defined in Example 1A?

An egg shell model has no wall thickness defined. In this Example, and assuming that the egg shell surfaces are drawn along the centerlines, the result is that all three: ShellGeometry, SurfaceBoundary, and Surface elements would all be identical. In other words, the SpaceBoundary and Surface elements would not change at all, but the Coordinates for the ShellGeometry PolyLoops would have the same values as their respective SpaceBoundary counterparts. In a sense, the ShellGeometry and its descendants just become redundant for egg shell models.

## Roof is Constructed with all insulation thickness below the Wall Height.

To show a wrinkle of what will happen if the roof insulation moves inward. The plan and axonometric view look nearly identical. The major difference between the two models can be determined by looking at Figure below and comparing to Figure 7 in example 1A:

roof thickness

(insulation entirely below the

top of the walls)

z = 9.5

z = 0

slab on grade

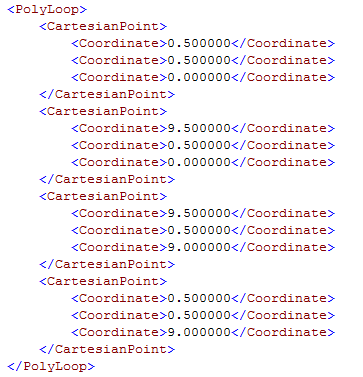
thickness

Figure : Section view of the geometry for Example 1B.

### gbXML Diffferences with Example 1A

The only major difference between this version and the previous is the definition of the ShellGeometry PolyLoop elements that define the surface area of the exterior wall’s interior wetted surface. There is no change to the SurfaceBoundary and Surface elements of the wall, floor, or roof. Also, the Space Volume will now be decreased due to the insulation now absorbing some of the interior volume of the space.

Compare the XML code snippet below to the one above for example 1A’s south wall.



Code Snippet : ShellGeometry PolyLoop for the South Wall in Example 1B.

The remaining definitions for the objects remain consistent with the definitions in 1A. The Z-coordinate of the roof element is now brought down to 9.5, but remains consistent across all elements, i.e. – the centerline of the roof is not defined.

What may raise more issue with those unfamiliar with the gbXML standard, is that the Surface and SpaceBoundary definitions for both walls and roofs remain the same. In other words, the roof dimensions are still 10’ x 10’ in gbXML, even though the dimensions of the roof have been reduced to fit within the confines of the walls. Also, the exterior walls are also declared as being 10’ tall, though now a fair percentage of the wall is actually backed by roofing material.

The justification behind this effect is simply that the gbXML file must create an enclosed shell and the extra surface area, in at least a general way, captures the complex 2-D heat transfer at the intersection of the wall and roof. Though not a perfect description in reality, this is the current interpretation of the situation as translated into gbXML.

### How would an egg shell model be defined in Example 1B?

This configuration cannot be modeled in an egg shell model, unless the thickness of the roof surface is modeled as a separate zone. It is not possible, with a single roof surface, to account for the thickness of the roof and its effect on both:

* the resultant change in the ShellGeometry roof PolyLoop coordinates
* the change in the Space Volume value

It is deemed, as of the writing of this document, that the current egg shell models will simply have either a different wall height or a different room volume when attempting to create the same given geometry. This is not to imply that the egg shell model is in any way less or more accurate than a model generated with construction thicknesses.

# Example 2 – Two Cubes of Equal Size Stacked Side by Side

## No Interior Hole or Opening in the Surface Separating the Two Cubes

Figure : A simple axonometric view of two cubes with wall thickness 'x', represented as Shell Geometry (thin blue solid line) Surface Boundary/Surface Element(thin blue dotted lines) and the exterior surface of the shape (thick red line). This diagram is to elaborate on how interior surfaces are represented in gbXML

interior wall that shares a common centerline

Figure : Plan view of same configuration as Figure 4.

Space -1

Space -2

Figure : Section view of same configuration as Figure 4.

This case is an adaptation of Example 1A, now with a second cube sharing a central partition with the first original cube. This center wall will become defined by the Surface attribute surfaceType as having the value “InteriorWall”. This example will highlight the basic changes that take place in the ShellGeometry, SpaceBoundary, and Surface elements and their respective descendants.

The ShellGeometry for each of the 2 respective Spaces elements in this example will each have six surfaces, and six PolyLoop elements, whose sequence of coordinates will allow for the formation of normal vectors via the right hand rule, similar in formation to Example 1A .

The SpaceBoundary is the first occasion where there is some indication that a surface is interior, and that there is a relationship between surfaces that share a common physical boundary. As shown in Figure 11, the internal partitions that separate Space 1 and Space 2 share a common centerline, it is in effect just one partition. This is the most common situation for interior walls that separate two spaces. Since SpaceBoundary elements represent the geometry at the centerline of walls, having a single centerline effectively means that there is really only one SpaceBoundary element, though two Space elements.

gbXML moves around this paradox by allowing the BIM authoring tool to define the same SpaceBoundary twice, in both the respective Space elements that share its boundary. In each instance of the definition, the SpaceBoundary element will have the same surfaceIdRef values in the same sequence. The PolyLoop Coordinate elements that are descendants of the SpaceBoundary should still be sequenced to form a proper right hand rule and outward normal that is pointed in the appropriate direction, which is contextual based on the order in which the surfaceIdRef are declared (see Fig 3). The Surface elements will therefore only contain a single instance of the SurfaceBoundary, since the value of any Surface elements’ id attribute must be identical to the value of the surfaceIdRef attribute of the SurfaceBoundary. Since the SpaceBoundary elements for interior walls share a common surfaceIdRef, it stands to reason that there can only be one Surface element.

For examples of ShellGeometry, SpaceBoundary, and Space element geometry definitions, see example 1A. Below are the definitions for the interior Wall. Assume the Space on the left is ‘Space-1’ and the Space on the right is ‘Space-2’.

……..



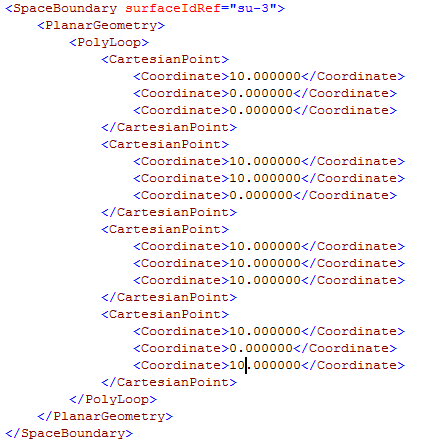
…….



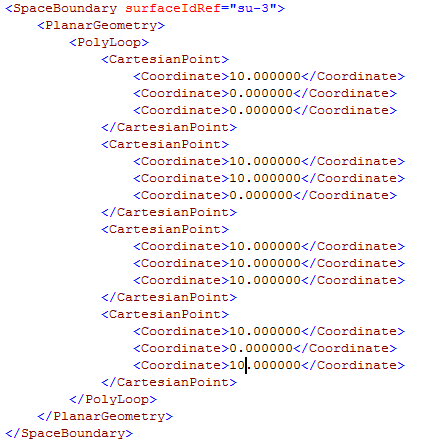
Code Snippet : Shell geometry definitions for interior walls, in each of the respective spaces that contain it

### Resulting PolyLoop Outward Normal Vectors for Shell Geometry

……



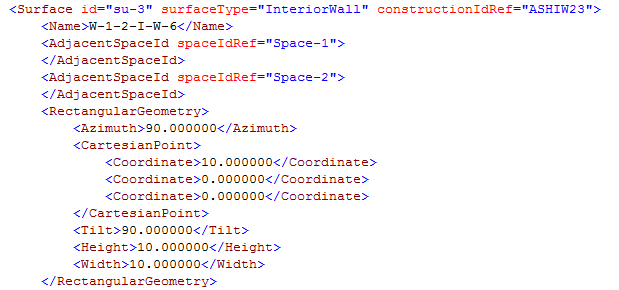
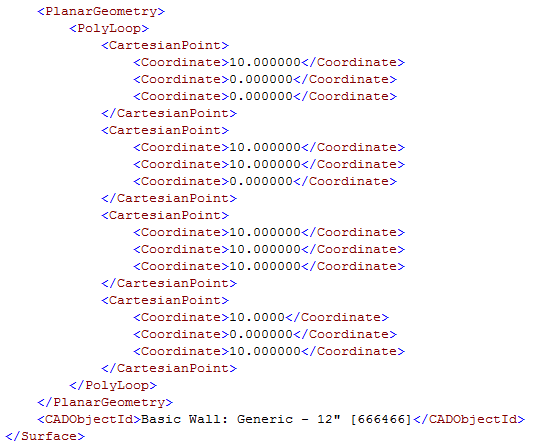
 ……



Code Snippet : Space Boundary interior wall definitions as described in both Space 1 and Space 2. Both SpaceBoundary surfaceIdRef values are identical.

Figure : The resulting outward normal vectors for interior walls as defined by SpaceBoundary and Surface elements (outward normal vector in red). The vectors are shown in axonometric(on the left) and plan view (on the right).

The resulting Surface element definition distills the SurfaceBoundary geometry definitions down to a simpler form. The Surface element for any interior wall will only appear once in the gbXML file, as opposed to the SurfaceBoundary occurring twice.



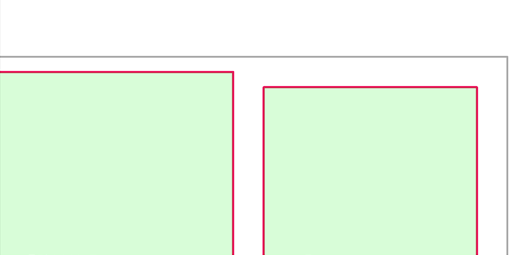
*Special Cases: Walls of different thicknesses*

Walls of different thicknesses typically have not been described by gbXML files. The thickness is of each wall is assumed constant in this section.

The ShellGeometry and SpaceBoundary nodes and their children are sufficient for describing the geometry for walls of different thicknesses, as these proofs reveal, though there is a minor loss in the model’s fidelity as will be described.

In the example figure below, a 6” wall is butted to a 12” wall with one of the faces flush. A second 12” wall (interior wall) is joined perpendicularly to these two.

x=0



y=20

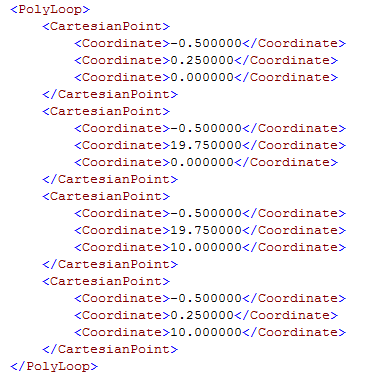
Figure : Two parallel walls of different thicknesses butted to one another with flush faces

The issue at hand is the small sliver where the 6” wall centerline does not intersect the centerline of the 12” wall, though they are parallel (circled in the above figure with a red dotted line). This raises a paradox in the gbXML description, which relies on a centerline description to characterize the position of the walls. How should the interior wall be described? Should it be described as extending to the centerline of the 6” wall, or the centerline of the 12” wall? What will happen to this small extra “sliver” that is present in the geometry description?

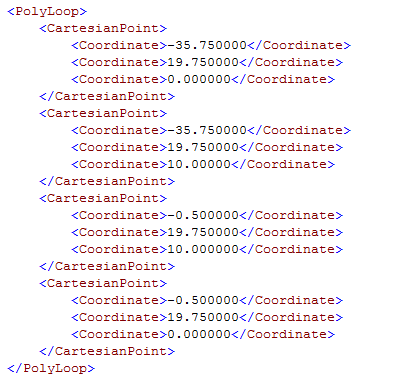
Sample gbXML is described in the following way

<ShellGeometry>

<ClosedShell>



12” interior wall



6” exterior wall

</ShellGeometry>

</ClosedShell>

For the enclosed region that defines the space bounded by this 6” exterior wall and 12” interior wall, everything remains virtually as would be expected. For the corresponding SpaceBoundary PolyLoops, they are defined in the following way.



6” exterior wall



12” interior wall

These examples highlight the paradox because closer examination of the SpaceBoundary (and hence Surface) element reveals that the interior wall has been described as having coordinates that coincide with the centerline of the *thinner* wall. This increases the size of the interior wall a small amount, but where thicknesses of two walls are not too dissimilar (most cases) this is a reasonable assumption.

Taking this convention removes the need to declare a sliver wall, which should only be present when the thicknesses of the walls deviate by an as-yet-to-be-agreed-upon difference. The use of the sliver wall was the previous convention for modeling these situations but is being revised to use this approach.

# Modeling Stacked Walls

Reserved for Future

## Modeling walls of different thicknesses that are Stacked on Top of one Another

Reserved for Future

# Modeling Roof Eaves

A roof may be drawn in the CAD/BIM authoring tool using a single planar element that extends beyond an external wall, forming an eave. This eave must be separated from the roof element that bounds the house, forming two surfaces. One is of surfaceType = “Roof” with an appropriate AdjacentSpaceId. The other, will be of surfaceType = “Shade”.

See the examples in Test Case # 28 for an example of proper gbXML.

# Modeling Walls that Originate Below-Grade and Terminate above-Grade.

In this case, a single wall must be broken into two walls, one that represents the condition where the exterior of the wall is exposed to the ground, and a second wall that represents the condition where the exterior of the wall is exposed to outside air and sun, with wind exposure. The BIM authoring tool shall be capable of making this distinction automatically, authoring gbXML that reflects this condition.

See Test Case 5 for an example.

# Interior Walls with Holes Cut for Passageways between Zones

It is common for interior walls on adjacent neighboring building floors to be stacked on top of one another, with potentially different locations for openings for egress, particularly in hospitals and modern complex office buildings.

See the full gbXML code provided in Test Case # 25 for an example of proper gbXML.

# Summary of Shell Geometry, Surface Boundary, and Surface Elements

## Summary of Important Features of the Shell Geometry and Surface Boundary Element

* ShellGeometry and SpaceBoundary Elements must define planar, non self-intersecting polygons.
* ShellGeometry and SpaceBoundary Elements, and their children, are used to accurately define the wall thickness of volume enclosures. This geometry definition can be used to accurately re-create the building geometry for simulations that benefit from an accurate geometric description of the wall thickness; (e.g. – daylight simulations in Radiance, CFD)
* Under all occasions, the ShellGeometry and SpaceBoundary of an enclosed volume in gbXML have coordinates that are defined along the interior wetted surface of the space and the centerline of the wall thickness, respectively. Even for stacked walls of different thicknesses, this should be sufficient for describing the wall geometrically, so long as the wall is simple and typical (90o tilt).
* Holes that may be created in the BIM authoring tool have no thickness. Therefore, the ShellGeometry and SpaceBoundary coordinates are identical in the case of holes.
* BIM authoring tools that create egg shell geometry have the same coordinates for ShellGeometry and SpaceBoundary Elements

## Summary of the Important Features of the Surface Element

* Surface Elements must define planar, non self-intersecting polygons.
* Surface elements have no wall thickness accurately defined in the element or its children in the gbXML file. Where the BIM authoring tool has a wall with a defined thickness, the Surface Element is defined at the centerline of the wall surface as it has been defined in the BIM authoring tool.
* The Surface Element thickness is instead defined by the construction associated with the element. The thickness of the wall drawn in the BIM authoring tool therefore, has no bearing on the wall thickness of the surface element. In other words, the Surface element is an approximation as drawn by the user in the BIM environment. It is assumed that this representation is “close enough” to reality that the use of Surface elements is justified.
* Surface elements are simplified representations of a wall, floor, or ceiling. The Surface element is essentially the same geometric information represented by the SpaceBoundary element, though in the case of interior surfaces, the height of the wall is re-defined to remove the thickness of the interior floor/ ceiling.
* Surface elements are closely related to SpaceBoundary elements, though represented in a different format that is more suited for consumption by many engineering simulation programs
* Surface elements, because they are not nested under a Space element in a gbXML file, must have some means to represent their relationship to the spaces that are adjacent to them. This is accomplished through the use of the Surface Element Child Node <AdjacentSpaceId spaceIdRef = ‘some space’ ></AdjacentSpaceId> where the attribute ‘spaceIdRef’ is equal to the ‘id’ attribute found in the Space element in the gbXML file.
* Each Surface Element has an attribute “id” which must be defined. This id is a string that is identical to the SpaceBoundary attribute “surfaceIdRef”
* Surface elements that have been generated by BIM authoring tools that define egg shell geometry, shall have geometric coordinates where the ShellGeometry planar surface has been drawn.

# Defining Openings

Openings should be defined in gbXML as children of Surface Elements. Loosely:

<Surface>

…..

<Opening>

….

</Opening>

</Surface>

Openings can refer to windows (both fixed and operable), skylights (both fixed and operable), doors, and “holes” or cut outs in a surface for daylight penetration or to improve ventilation air movement. The enumerations for Opening objects are:

Many of the facets of the opening definition can be found in the XSD. Structurally, they are very similar to surfaces and contain much of the same information. All of the information required for geometry is housed in two basic elements, the RectangularGeometry and the PlanarGeometry elements, and their descendants.

## Essential Opening Element Rectangular Geometry Descendants for Validation

### Azimuth

This is available but should be unused. It should always read 0 (zero for clarification, not the letter “O”). If a user or the validator wishes to know the actual azimuth of the Opening, refer to its parent Surface

### Tilt

This is available but should be unused. It should always read 0 (zero for clarification, not the letter “O”). If a user or the validator wishes to know the actual tilt of the Opening, refer to its parent Surface

### Cartesian Point

The Cartesian point is to be defined only in two dimensions. Though there is a Z-coordinate, it is not used and should always read 0 (zero for clarification, not the letter “O”). The two coordinates that are given in X and Y are the distance in X and Y from the lower left corner of the Surface to the lower left corner of the Opening

Figure : Definition of the Opening Insertion Point. The X and Y distance shown is the distance between the Surface Insertion Point (blue dot) and the Opening insertion point (red dot).

y

x

### Height

Height may be given but also may not be useful if the Opening is not a square or rectangle. The validator makes the assessment whether height is a useful value to check and if so, it will check the value given here. Otherwise, it is ignored.

### Width

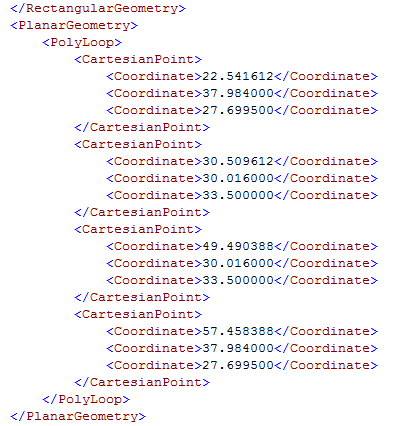
See Height above.

### PolyLoop

This PolyLoop is not used, in lieu of the Planar Geometry PolyLoop described below.

## Essential Opening Element Planar Geometry Descendants for Validation

The PlanarGeometry element houses only a single PolyLoop for each Opening, that has no bounds on the number of CartesianPoints allowed. This PlanarGeometry PolyLoop must meet all the requirements of any other PolyLoop, the coordinates must form a planar, non-intersecting polygon. The coordinates are defined in the global reference frame, as other Coordinates for ShellGeometry and SpaceBoundary elements. The Coordinate values are NOT local to the surface, as in the PolyLoop that may be defined in the RectangularGeometry element



1. As of this writing, version 5.10 [↑](#footnote-ref-1)
2. version 5.10 as of this writing <http://gbxml.org/currentschema.php> [↑](#footnote-ref-2)
3. version 5.10 as of this writing <http://gbxml.org/currentschema.php> [↑](#footnote-ref-3)